REMARKS

Claims 27-35 and 38-51 stand rejected under 35 U.S.C. § 103 on the basis of Faraboschi, et al. '576. Claim 36 stands rejected under 35 U.S.C. § 103 on the basis of Faraboschi, et al. '576 and Tannenbaum. Applicants respectfully traverse these rejections because the Examiner's reliance on Official Notice is misplaced, and there is no motivation or suggestion to modify Faraboschi, et al. by storing imaginary address information in a compressed program together with compressed form instructions, as in the independent claims of the present invention.

The Examiner acknowledges that Faraboschi does not teach that imaginary address information should be stored in the compressed program together with the compressed-form instructions, and Applicants agree. In fact, Faraboschi adopts a fundamentally different address management approach to that of the present invention.

As shown in Faraboschi Fig. 2, the main memory 110 in which the compressed program is stored has a code pointer segment 130 and a code heap segment 140. The code heap segment 140 stores exclusively compressed instructions. The code pointer segment 130 stores a series of code pointer words 132, 134, 136. Each code pointer word comprises a 9-bit mask, which is similar to the decompression key of the present invention, and a 23-bit code pointer (ptr)152. The code pointer 152 may be regarded as being stored in the program memory *together with* the compressed-form instructions, as it appears that the code pointer segment 130 and code heap segment 140 are all part of the same memory 110. However, the

pointers are not used by the processor to allocate assigned imaginary addresses to the decompressed instructions, nor are they capable of being used for this purpose (which is the purpose specified by the independent claims), as will now be explained.

In the address management method adopted in Faraboschi (Fig. 2), there is a one-to-one correspondence between addresses in the code pointer segment 130 and lines in the instruction cache (column 4, lines 55 to 57). The processor's program counter PC accordingly steps through successive addresses in the code pointer segment 130 and uses the code pointers 152 to identify the physical addresses in the code heap segment 140 at which the compressed-form instructions for successive cache lines are stored. Accordingly, in Faraboschi's implementation, contrary to the requirements of the present independent claims, the address information which is stored in the program memory (i.e. the set of code pointers 152) does not represent any imaginary addresses at which the instructions should be considered to exist when held in decompressed form in the instruction cache 100. Instead, the stored address information (set of code pointers 152) is merely used to map from an imaginary address *already known to the processor* to the physical address in the code heap segment at which the relevant instruction is stored. The program counter PC itself directly represents the imaginary addresses because the address in the code pointer segment that is specified by the program counter is the same as the imaginary address at which the instructions are considered to exist when held in decompressed form in the instruction cache 100 (see the cache tags in Faraboschi Fig. 2). Indeed, Faraboschi explicitly states (column 4,

lines 53 to 55) that "the code pointer words are located at the same instruction addresses as corresponding instruction words in instruction cache 100".

The address management method in the present invention is effectively the opposite of that used in Faraboschi Fig. 2. In the present invention, the program counter PC points directly to the location in the program memory at which the next section of compressed-form instructions is stored. The processor reads the imaginary address information from one of the sections, or optionally from each of the sections, and uses this information to assign imaginary addresses to the decompressed instructions. Thus, in the present invention, the imaginary address information is stored in the program memory and retrieved from it using a program counter which points to the compressed sections one after the next, whereas in Faraboschi Fig. 2, the imaginary addresses originate inside the processor and are exported from the processor (in the form of the program counter) to access the code pointers that are needed to locate the compressed-form instructions.

The Examiner alleges in paragraph 7, that "address information could easily be stored with the instruction [in Faraboschi] to allow for faster look-ups in the memory just like a cache tag" and that this would "enable easy setting of the cache tag when loading the cache with instructions from the memory by copying the information into the cache tag". However, as indicated above, the address management method in Faraboschi is to have the program counter specify the imaginary address, so that the cache tag is simply the upper 20 bits of the program counter. There would therefore be no motivation in Faraboschi to store imaginary address information with the instructions. Furthermore, in the Faraboschi address

management scheme, no useful purpose would be served by including in the program memory (either in the code pointer segment 130 or in the code heap segment 140) imaginary address information based on the sequence of the original instructions prior to compression. This imaginary address information could only be retrieved from the program memory by using imaginary addresses already specified by the program counter. In other words, if the program counter did not already point to the correct imaginary address, it would not be possible to use it to locate any imaginary address information stored in the program memory.

The Examiner's reliance on Judicial Notice is inapposite. Merely because references such as Yoshida contain examples in which addresses are associated with instructions does not make it obvious to store imaginary address information of the kind specified in the independent claims, with compressed-form instructions to enable a processor to allocate the imaginary addresses to decompressed instructions during execution. This feature is not taught or suggested by Faraboschi either alone or in combination with alleged common general knowledge in the art, nor is it taught or suggested by Faraboschi in combination with Yoshida or any of the further references cited by the Examiner in paragraph 45 of the Office Action.

The distinction between the prior art in the present invention is a significant difference because with the present invention, a processor can perform decompression of a compressed program "on the fly", i.e., during execution of the program. The present invention solves a problem not addressed by the cited references, even considering the Examiner's Official Notice, and addresses a long-felt need, i.e., decompression of a

compressed program on the fly in an inventive way. For these reasons, withdrawal of the outstanding rejections is respectfully requested.

For the foregoing reasons, Applicants believe that this case is in condition for allowance, which is respectfully requested. The examiner should call applicants' attorney if an interview would expedite prosecution.

Respectfully submitted,

GREER, BURNS & CRAIN, LTD.

By 

Patrick G. Burns
Registration No. 29,367

September 28, 2005

300 South Wacker Drive
Suite 2500
Chicago, Illinois 60606
Telephone: 312.360.0080
Facsimile: 312.360.9315

Customer No. 24978